WAVEWATCH III ® basics

Installing and running the model

Hendrik L. Tolman
Chief, Marine Modeling and Analysis Branch
NOAA / NWS / NCEP / EMC

Hendrik.Tolman@NOAA.gov

Overview

- What is WAVEWATCH III.
- How to install the model.
- How to run the model :
  - Basics for single-grid model.
    - Including traditional one-way nesting.
  - Basics for multi-grid or mosaic model.
    - Including full two-way interactions between an arbitrary number of grids.

Background material

- Background information can be found on the website below, particularly recommended are:

    The manual.

    The best-practices guide for programming for WAVEWATCH III.

http://polar.ncep.noaa.gov/waves/wavewatch/wavewatch.shtml

- Applications can be found on the old and new NCEP operational wave model web sites below. Recommended information on these web sites are:

    The primer on the old web site.

    Two COMET training web sites on the new web site.

http://polar.ncep.noaa.gov/waves

http://polar.ncep.noaa.gov/waves/index2.shtml

Third generation wave model of WAM descent, but with many differences :

- Governing equations (action versus energy).
- Grid system (wavenumber on variable grid versus relative frequency).
- Numerics (dynamic time stepping, higher order propagation schemes, GSE alleviation techniques).
- Physics:

  Framework with multiple options.

WAVEWATCH I and II developed at Delft and NASA/GSFC based on different governing equations.

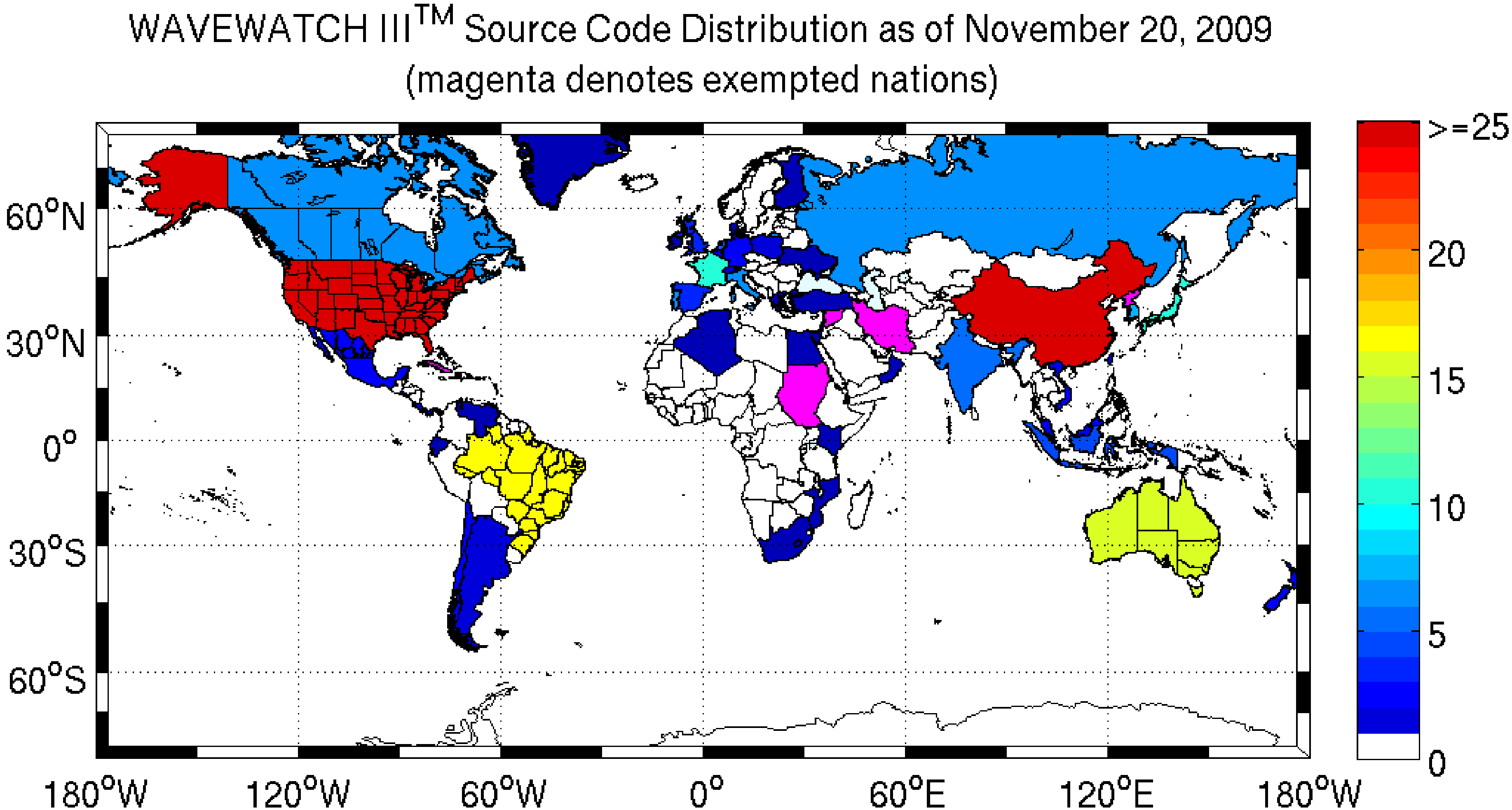
Similar to SWAN, but also with differences :

- Basis numerical solution techniques:

  WAVEWATCH III traditional hyperbolic equations solved by marching in time.

  SWAN (quasi-) stationary elliptical equations, solved by iterative sweeping of domain (even in unsteady model mode).

Model version 3.14 trademarked and licensed, with license as close to open source are US government allows.



WAVEWATCH III™ Source Code Distribution as of November 20, 2009
(magenta denotes exempted nations)
60°N
30°N
0°
30°S
60°S
180°W
120°W
60°W
0°
60°E
120°E
180°W
>=25
20
15
10
5
0

Distribution methods:

- Most WAVEWATCH III users will get the code from our web site after agreeing to the license.

  Tar files + install script:

  Manual chapter 5.
- Some who work as co-developers have access to our subversion (svn) server to get the latest developmental versions of the code.

  svn server + install script.
- We will focus on first distribution method, transitioning to second is trivial.

- NOTE: WAVEWATCH III is Linux/UNIX only. Installation on windows will require preparations using Linux/UNIX.

Distribution files:

install_wwatch3 (install_ww3_svn)	Installation script.
wwatch3.aux.tar	Auxiliary programs and scripts, including GrADS scripts.
wwatch3.ftn.tar	Source code files.
wwatch3.inp.tar	Example input files, identical to those printed in the manual.
wwatch3.tst.tar	A large variety of test cases.

Installation procedure:

- Copy five files to WAVEWATCH III designated directory, for instance ~/wwatch3
- Assure that install_wwatch3 has execute permission.
- Execute the install script and answer all questions.

  Will practice this in afternoon session.

  Will require a basic FORTRAN 77 compiler to be assessable, typically f77 (gnu) will do. This compiler will be used for aux programs only, not for actual WAVEWATCH III code.
- Add directories to search path in shell as directed by script.

Cont'ed

- NOTE: installation will generate file *.wwatch3.env* in the home directory.
- This file is used by all WAVEWATCH III management scripts, and points to directories used for code etc.

- HINT: If multiple versions of WAVEWATCH III are maintained simultaneously, then:
  - Point to proper directory by modifying .wwatch3.env by hand or by re-running install_wwatch3, or
  - Put a generic name like wwatch3 in .wwatch3.env and use this as a symbolic link to the actual wave model directory (my preference).

Compiling the code:

- WAVEWATCH III is not distributed as ready-to-compile FORTRAN 90 code, but has a set of scripts to build the model according to user specifications:
- Critical files needed to compile:

switch	List of model options selected by user (manual section 5.4). Preset with default model options.
comp	Compile script (section 5.3). Requires user interventions once.
link	Link script (section 5.3). Requires user interventions once.
w3_make	Compiles wave model code-by-code

Cont'ed:

- The compile and link scripts need to be modified to address error capturing for the given hardware and compiler.
- Set-up procedure described in manual section 5.3.
- Various prepared comp and link scripts are provided with the model distribution.
- Please provide us with yours for further distribution with the code

- NOTE: the compiler used here can be different from the compiler set in *.wwatch3.env*, compile optimization is set in these scripts.

- More details on model compilation follow in wwws 3.3.

Cont'ed:

- Installation on SMP or MPP system gets a little more complicated. Generally only the main program ww3_shel or ww3_multi will be run as a parallel code, whereas all other codes remain serial. Proper compilation requires one of the following options:

  First compile all auxiliary programs with the proper compile and link option of serial codes. Then reset compile and link options, for recompilation of all shared subroutines by running w3_new, and then compile ww3_shel alone.

  Generate individual and complete source codes for all programs and create the proper corresponding compile protocols.

Windows installation:

- WAVEWATCH III is not set up for installation under MS Windows®.

- For installation under windows, the following procedure can be used:

    Find a Linux/UNIX box and perform the basic installation.

    Set required model options in the switch file.

    Run the script w3_source to extract the clean FORTRAN codes and corresponding makefile in tar files.

        Set compiler options for windows compiler in w3_source, or

        manually edit makefiles as needed.

Model version 3.14 will be the considered here.

The model consists of a set of programs :

- Grid preprocessor (*ww3_grid*).
- Initial conditions program (*ww3_strt*).
- Model input preprocessor (*ww3_prep*).
- The actual wave model (*ww3_shel*, *ww3_multi*).
- Output post-processors:
  - Numerical (*ww3_outp*, *ww3_outf*, *ww3_trck, ww3_grib*).
  - GrADS (*gx_outf, gx_outp*).

Introduced in version 3.14

Introduced in version 2.22

Basics :

- All programs read their instructions from a a file with the extension ***.inp***, for instance ***ww3_grid*** reads from ***ww3_grid.inp***.
- The first character of the first line of the ***.inp*** file identifies the comment character.

  Lines starting with this character are skipped while reading.

  In all examples used by us the comment character is ‘$’.
- All programs write output to standard output.
- All raw output files of the wave model are identified by the extension ***.ww3***. (exception: running ***ww3_multi***).

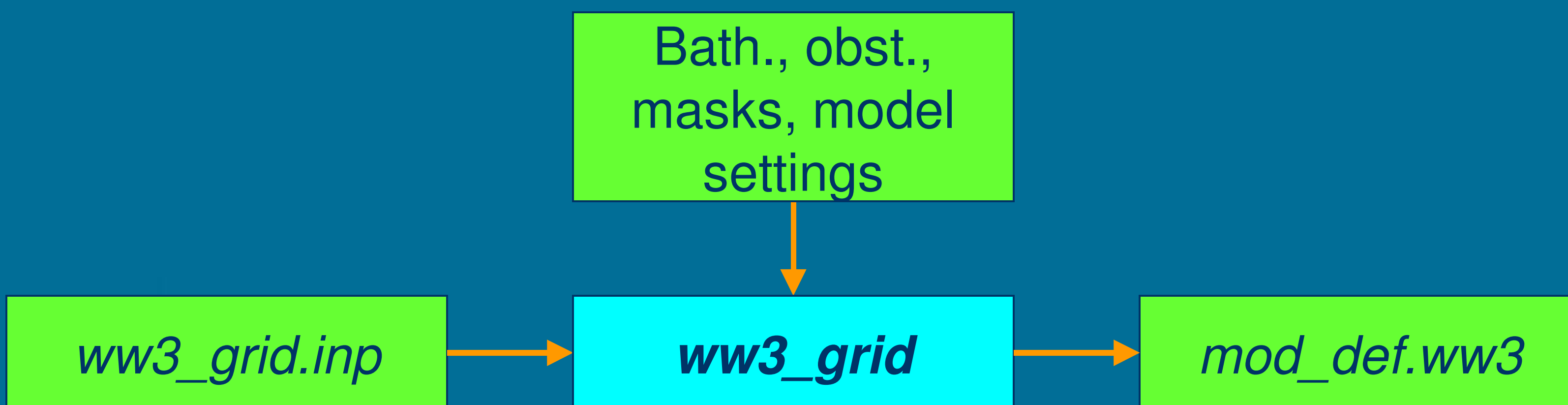
- The presentation will first focus on running the conventional single-grid wave model ***ww3_shel***.

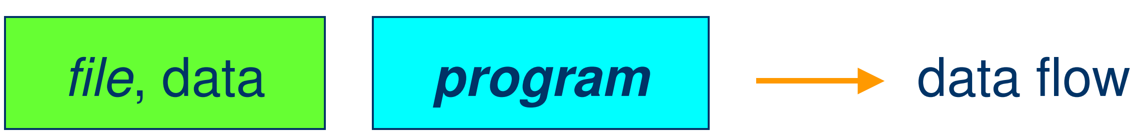
Cont'ed :

- Before anything else can be done, a model or grid is defined using ***ww3_grid***, resulting in a model definition file ***mod_def.ww3***. This file contains:
    - Bathymetry, obstruction grids, masks.
    - Parameter settings for all physics and numerical approaches (including flags and dry-run option).
    - Data fields for above approaches as needed.
    - Time steps for grid.
    - Input boundary points for grids.
    - Output boundary points for nested grids if off-line nesting is performed (***ww3_multi*** does in-line nesting).
        - Points are defined, but output is controlled by ***ww3_shel***.

Running ww3_grid :

- ***ww3_grid*** is typically run once, and ***mod_def.ww3*** is then stored for use with all other programs.





Creating initial conditions :

- Initial conditions are generated using ***ww3_strt.*** Options:

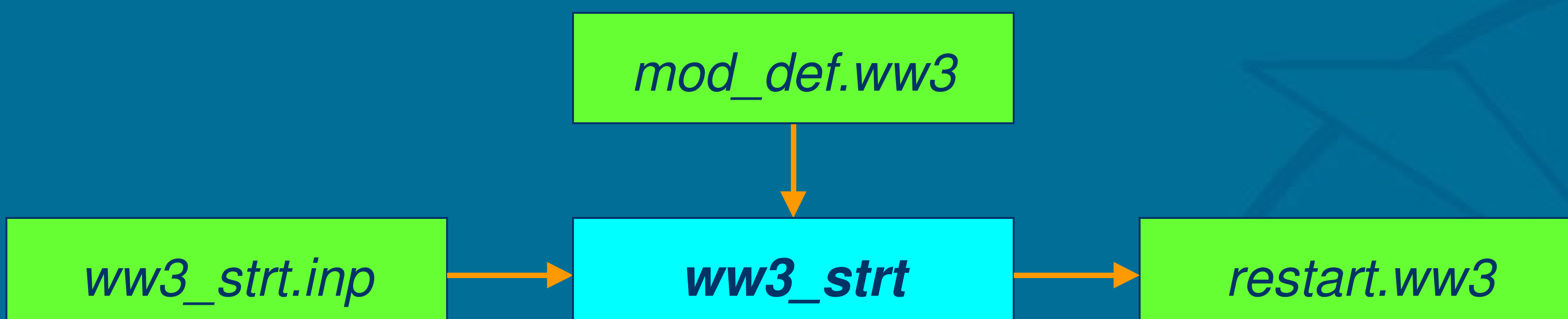
  Gauss / cosine (idealized swell).

  JONSWAP, Gauss in space.

  JONSWAP, fetch limited based on local wind and grid spacing.

  Single user-defined spectrum.

  All at rest (new in 3.14).



Cont'ed :

- Preprocessor ***ww3_strt*** and/or file *restart.ww3* not mandatory. If the file is not found, the main program will assume:

  Initializing with local wind if no linear growth option is selected.

  Otherwise start from rest.

- Note that the initial condition program can be used to set steady-state boundary conditions by defining points for input boundary data but by not providing a file with such data.

With the ***mod_def.ww3*** and ***restart.ww3*** files we have enough to run a basic wave model.

The input file for the basic wave model (***ww3_shel.inp***) contains :

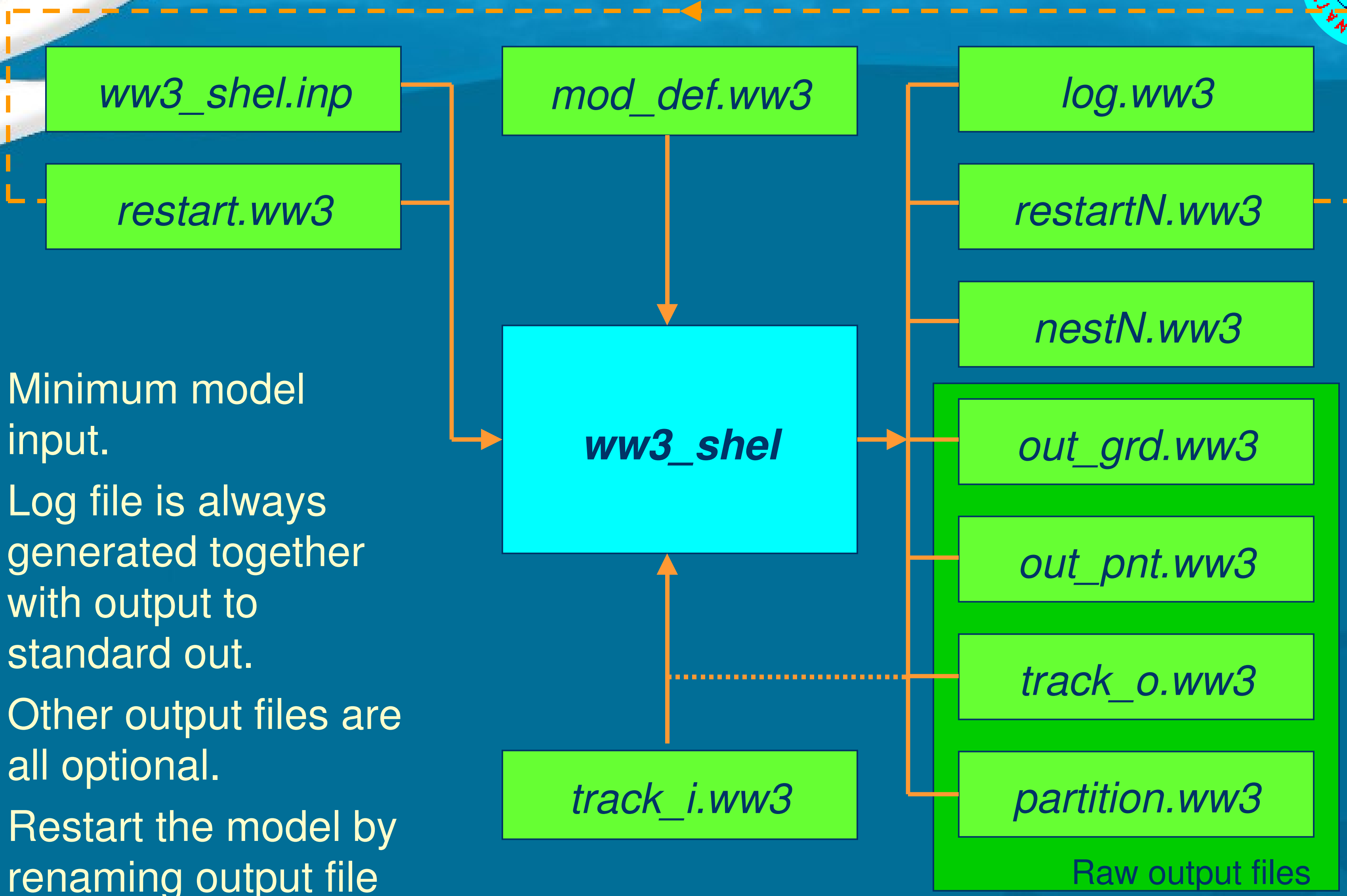
- Definition of inputs to use (level, current, wind, ice; homogeneous or variable).

  Homogeneous input data makes it possible to run model without separate input data files.
- Start and end time.
- Output requests for six types of standard output.
- Homogeneous input field data if requested.

ww3_shel.inp cont'ed:

- Output needs to be selected in ***ww3_shel.inp***.
    - Gridded output fields.
    - Point output containing full spectral data.
    - Nesting data for higher resolution grids.
        - Only output times set in ***ww3_shel.inp***.
        - Up to nine different files.
    - Restart files.
        - Up to nine different files.
    - Output of spectra along pre-defined tracks.
    - Partitioned spectral data (new in 3.14).

Running the model



- Minimum model input.
- Log file is always generated together with output to standard out.
- Other output files are all optional.
- Restart the model by renaming output file to input file.

Model output

- The ***ww3_shel*** model generates two formatted output files.
  - std out output tracks progress “on screen”.
  - ***log.ww3*** is generated by the actual wave model routines. This file is provided to assure consistent info on the model when the routine is included in an integrated model.

- The raw data files and the output restart file are only generated upon request (***track_i.ww3*** only needed for request). They are unformatted and partially packed.
  - ***restartN.ww3*** and ***nestN.ww3*** are used by subsequent model runs as is (after renaming).
  - All other raw output files require postprocessing.

- The file out_grd.ww3 contains fields of mean wave parameters such as the significant wave height, and is processed using ww3_outf. ww3_outf produces :

    File inventories of out_grd.ww3.

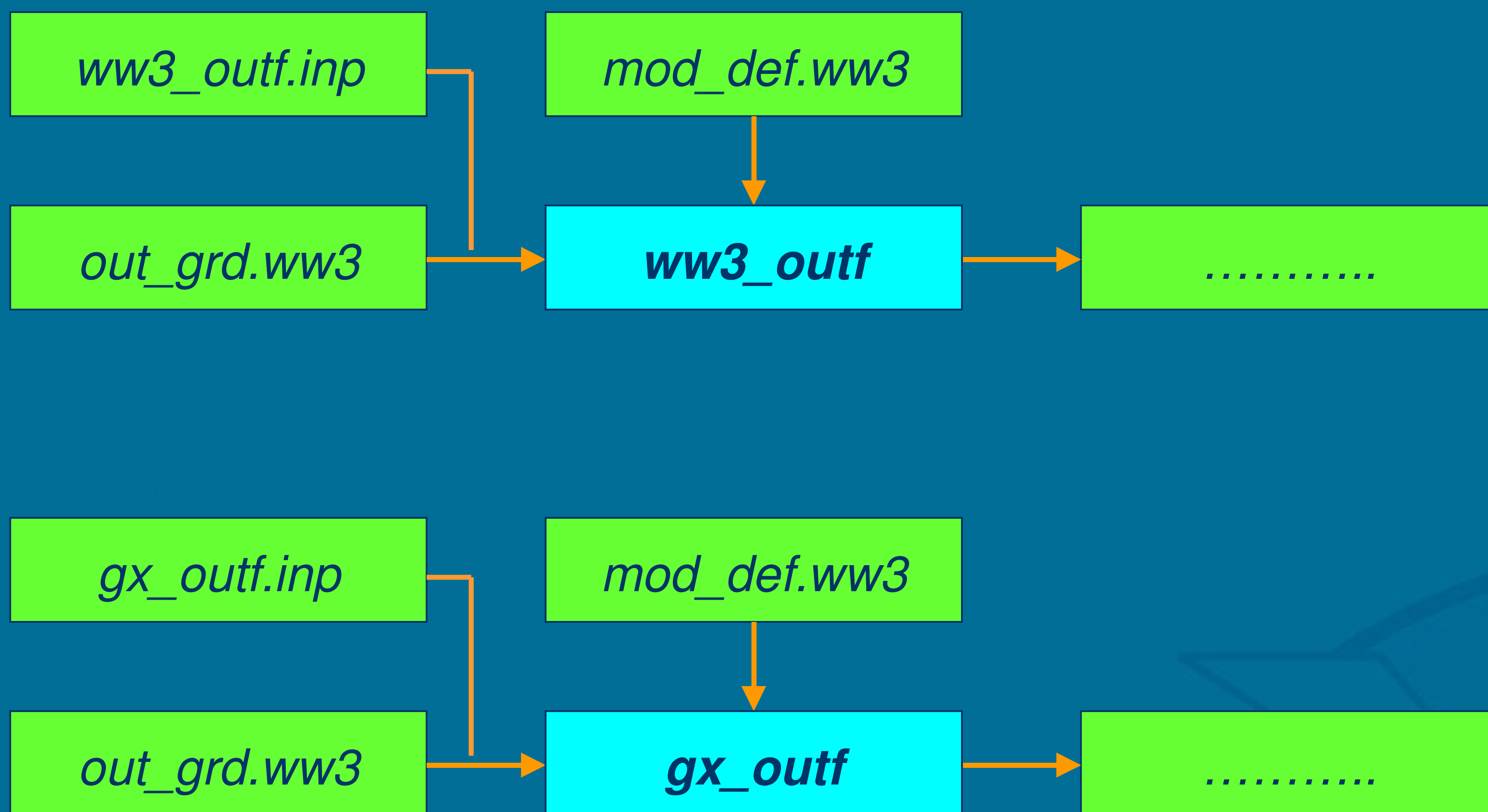
    Print plots of fields.

    Statistics for given ranges of the grid

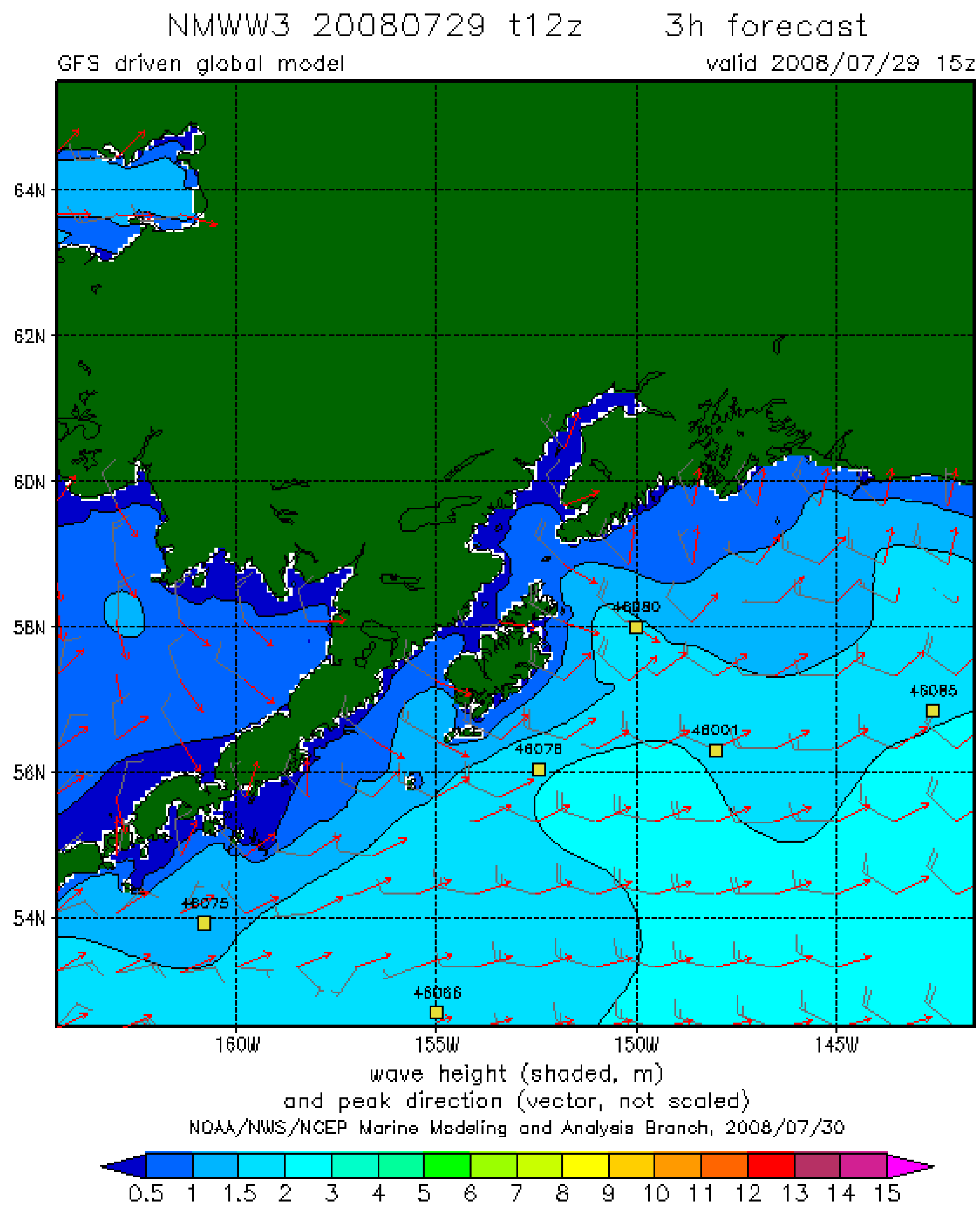
    Data transfer files.

- The file out_grd.ww3 can be converted into GrADS data files using gx_outf, which operates similar to ww3_outf.

Model output



Model output

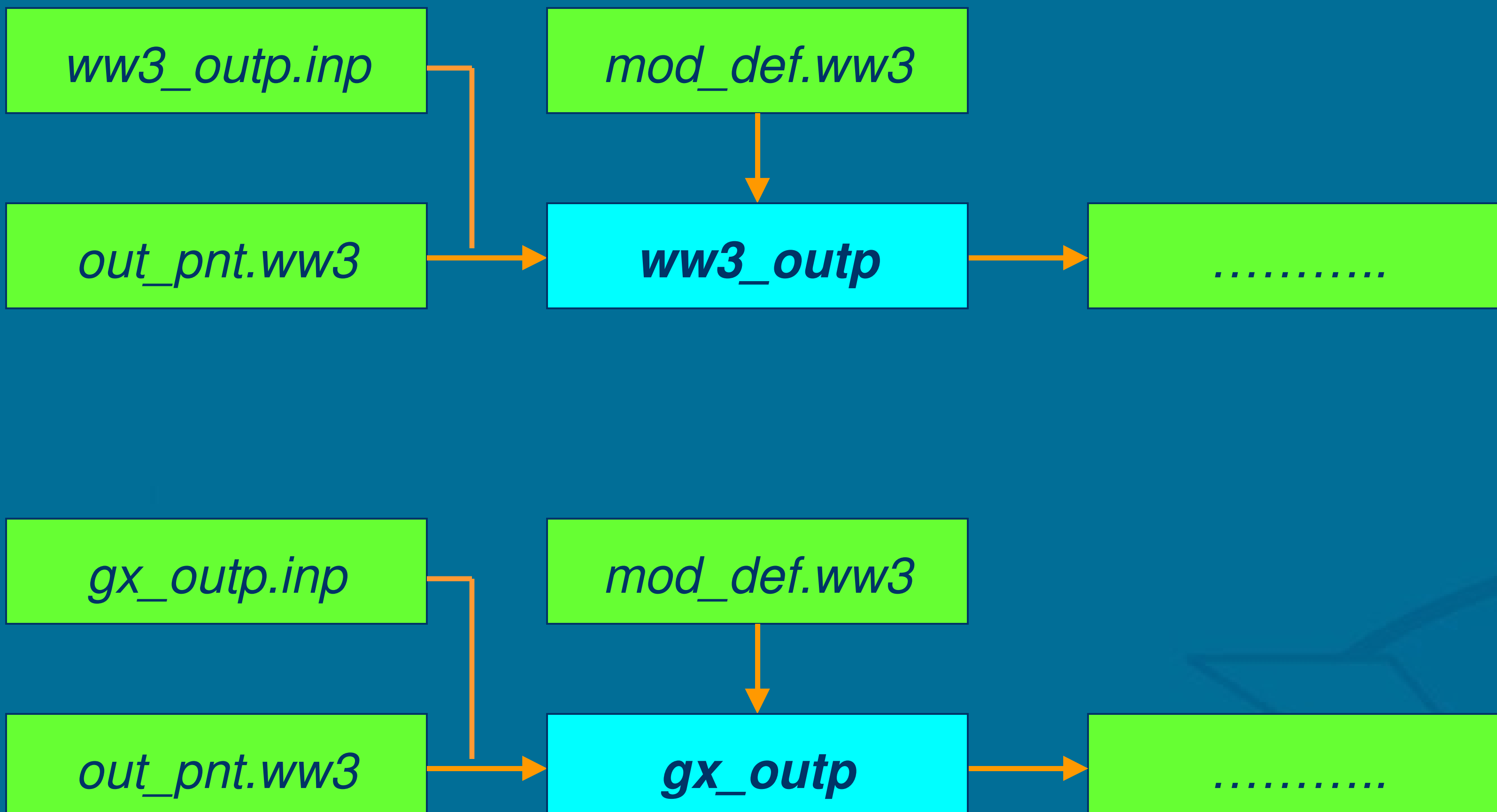


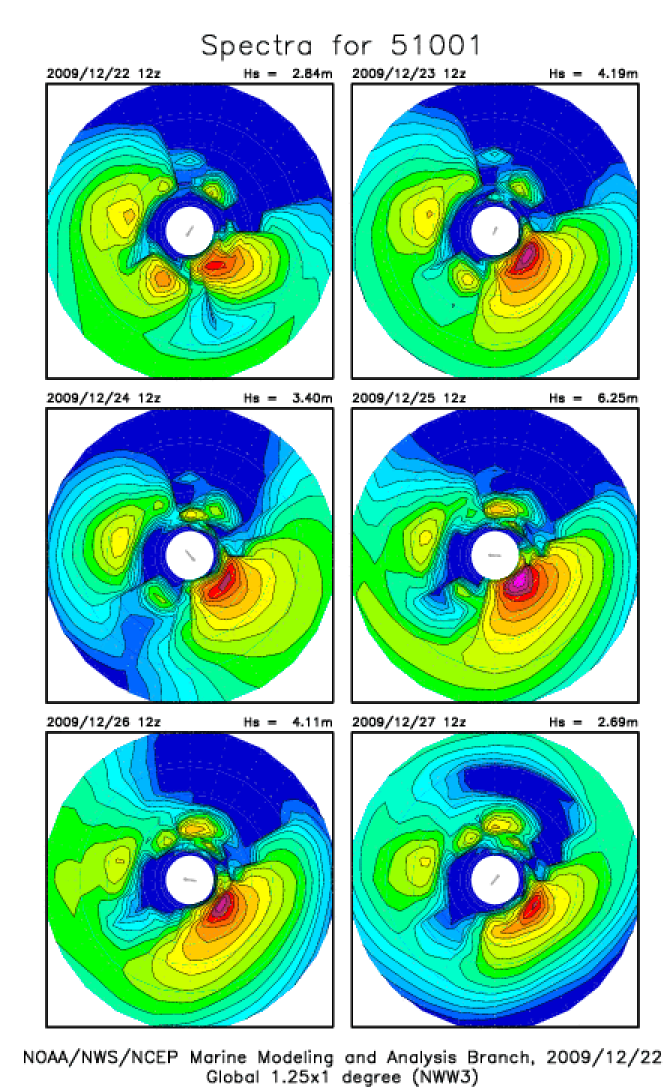
Example of GraDS output for present operational resolution for Cook Inlet in Alaska, generated after post-processing ***out_grd.ww3*** with ***gx_outf.***

Model output

- The file out_pnt.ww3 contains full spectral data for selected output points, and is processed using ww3_outp. ww3_outp produces :
  - File inventories of out_pnt.ww3.
  - Spectral data as print-plots, tables of 1-D spectra and data transfer files.
  - Tables of mean environmental and wave parameters.
  - Source term data similar to that of spectra.
- The file out_pnt.ww3 can be converted into GrADS data files using gx_outp

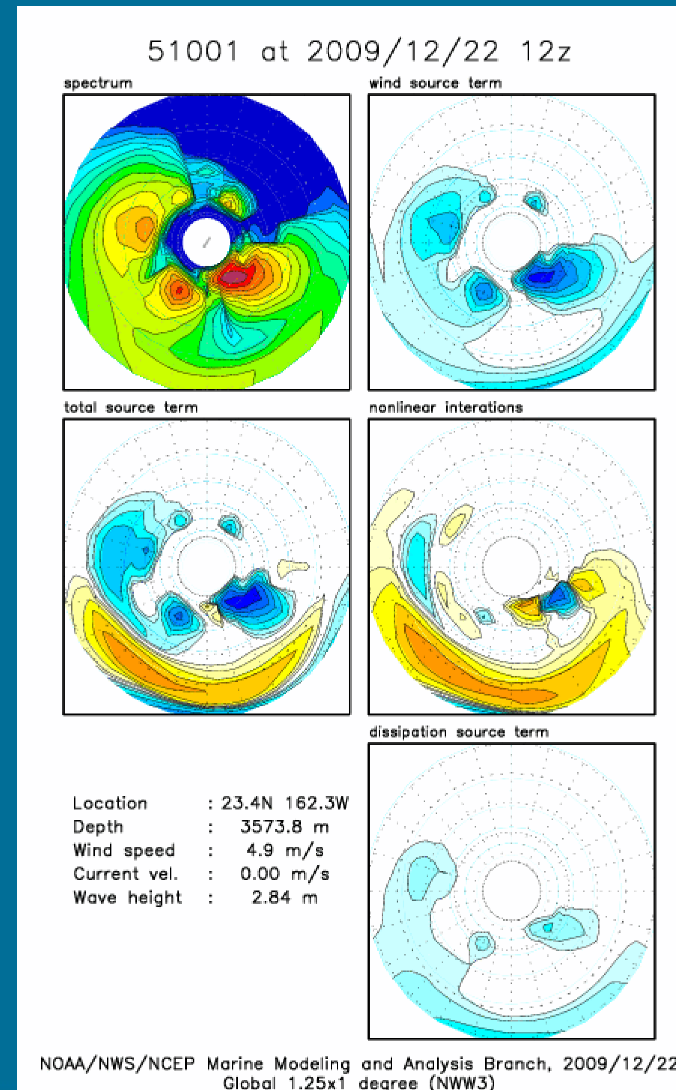
Model output



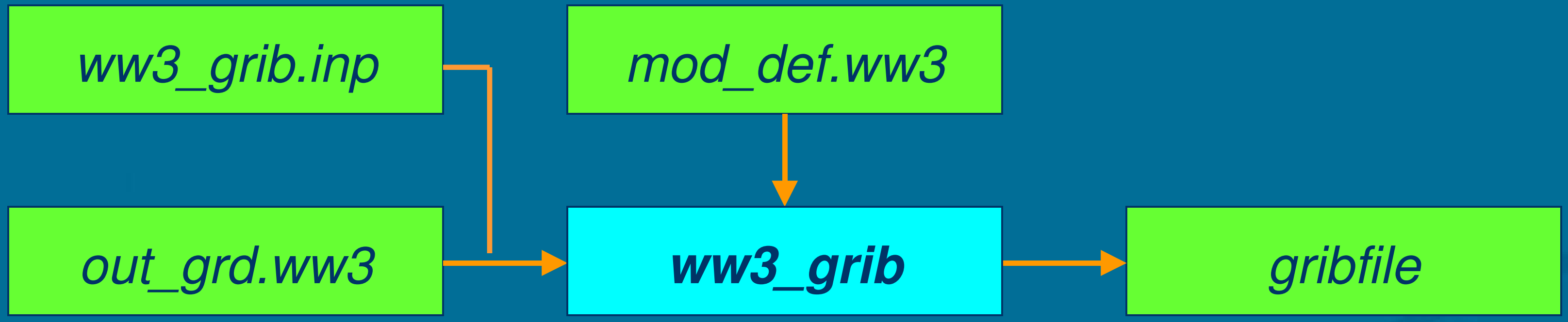


Model output

Example of GraDS spectral and source term output made by processing ***out_pnt.ww3*** with ***gx_outp***, and using GrADS script provided with the wave model distribution.

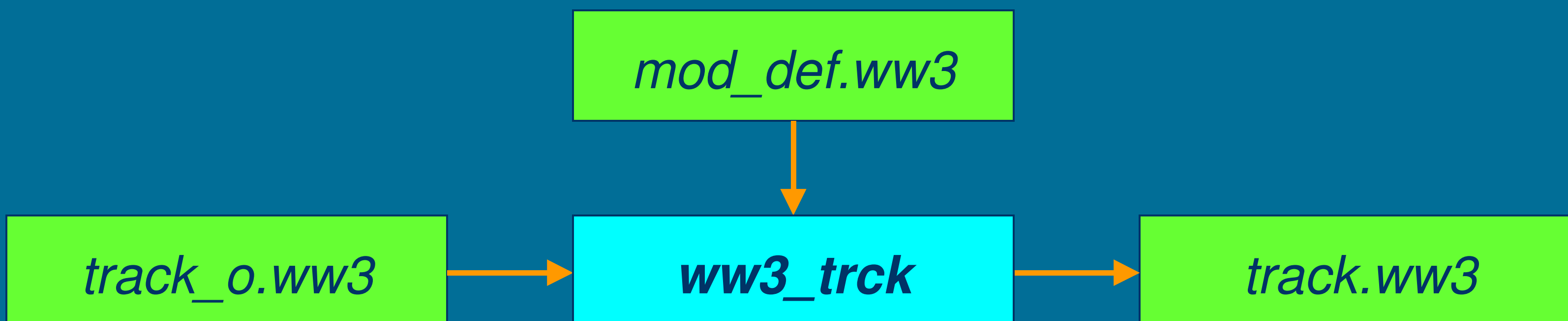


- ***ww3_grib*** takes the raw data file ***out_grd.ww3*** and converts it to a GRIB or GRIB2 file.
- GRIB and GRIB2 packing routines are not provided with WAVEWATCH III yet.



Model output

- ***ww3_trck*** takes the unformatted, direct access file ***track_o.ww3***, and repacks it in a free format readable, integer packed data file ***track.ww3***. No file with command data is needed.



So far, we have been running the model with homogeneous input fields for

- water levels
- currents,
- winds,
- and ice.

None of these input fields are mandatory, and a mixture of homogeneous on non-homogeneous fields can be used.

All input fields are pre-processed and written to files with a standard WAVEWATCH III format.

- This separates processing input from the wave model.

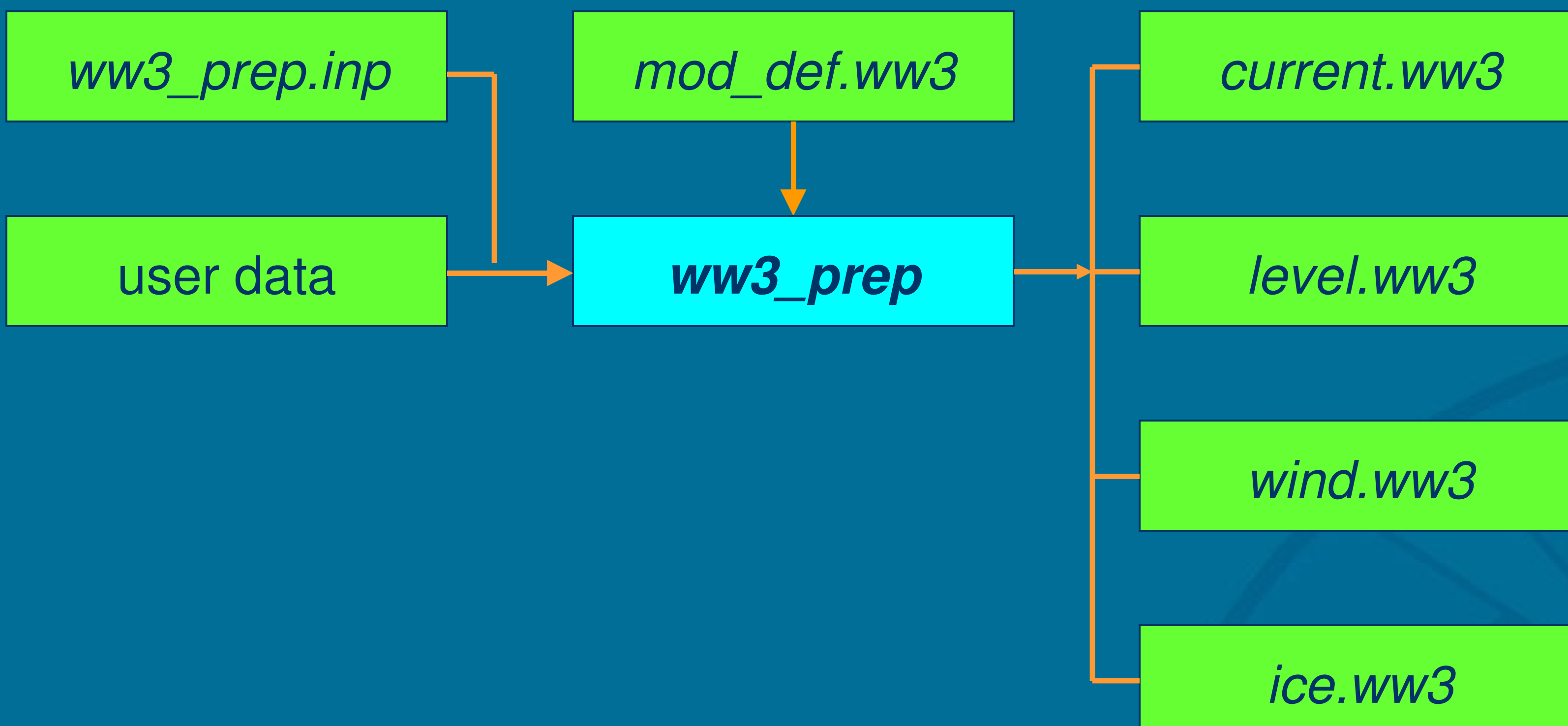
In the standard WAVEWATCH III data files, each data field has its own time stamp. ***ww3_shel*** deals with these time stamps on a field by field basis. This implies that :

- The model does not require input fields at constant intervals.
- The user need not supply field interval time steps in any of the programs as they are meaningless.

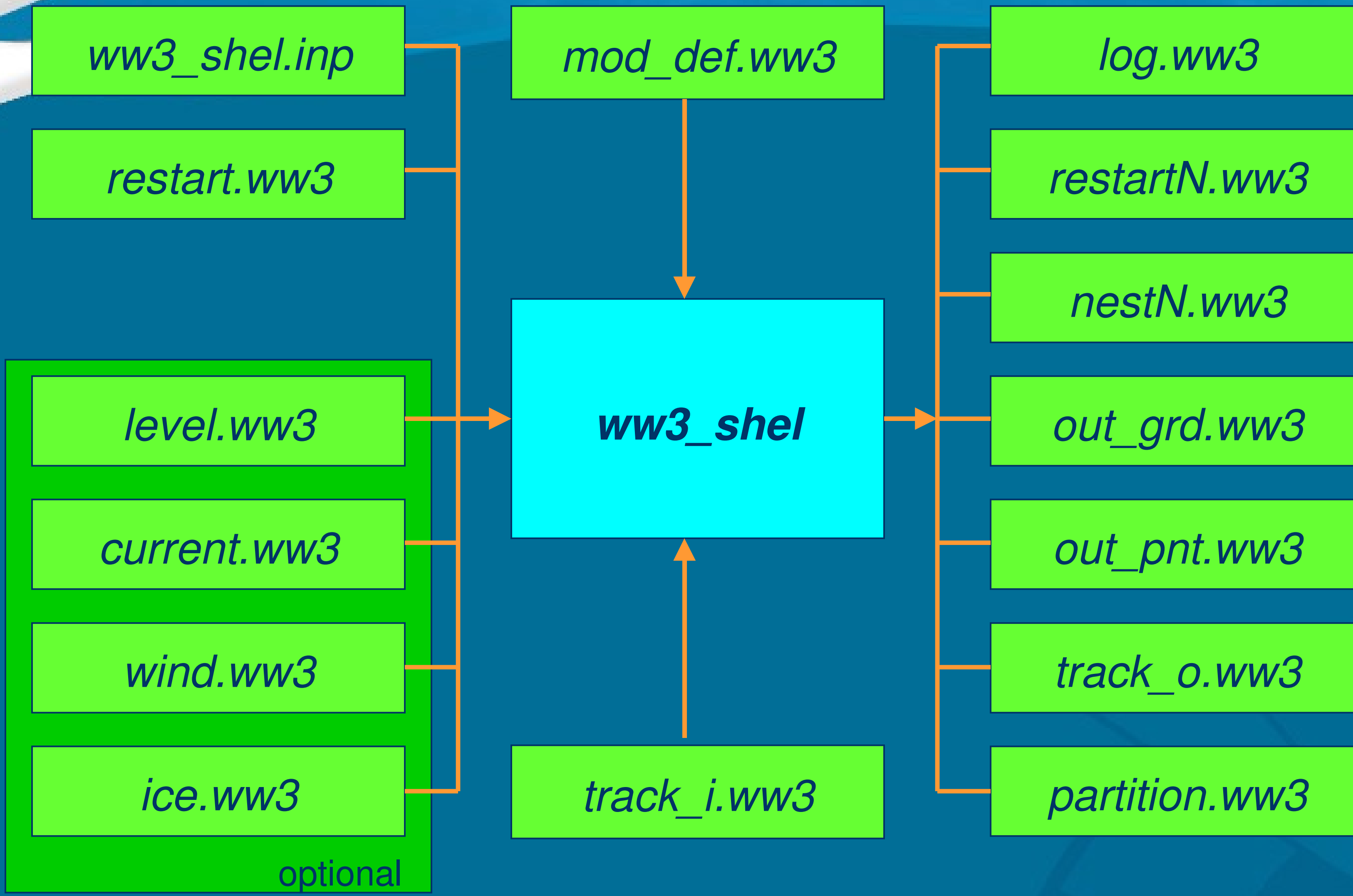
Input data are kept constant before the first and after the last time stamp in the file.

Model input

- Preprocessing of input fields is performed by ***ww3_prep***. This program deals with all four types of input fields (one at a time) and can deal with user-supplied fields of several formats.



Model input


ww3_shel.inp
restart.ww3
level.ww3
current.ww3
wind.ww3
ice.ww3
optional
mod_def.ww3
ww3_shel
track_i.ww3
log.ww3
restartN.ww3
nestN.ww3
out_grd.ww3
out_pnt.ww3
track_o.ww3
partition.ww3

- The user might need to repack his/her fields in order for ***ww3_prep*** to be able to read it. For instance, there is no GRIB or NetCDF option in the code for reasons of portability.

- HINT : Use the dry run option in ***ww3_grid*** to run all input fields through the wave model without doing actual wave calculations. This gives you a change of checking the data flow without having to waste expensive clock cycles.

Nesting (ww3_shel)

- Nesting in ***ww3_shel*** (as in most wave models) is a one way street. Boundary data is generated by a large scale model and read by a small scale model.
- Presently, up to nine individual sets of boundary data can be generated by a single run.
- Data transfer for nesting is set up in ***ww3_grid(.inp)***.
- Execution of nesting depends on time step requests in ***ww3_shel.inp***, and availability of the file ***nest.ww3*** for the small scale run.
- If the file ***nest.ww3*** is not found, boundary data is kept constant at designated boundary points. This can be a handy feature for steady-state tests.

Nesting can be set up in the following way :

- Set up a large scale grid and a small scale grid using ***ww3_grid*** with separate input files ***ww3_grid.inp***.
- Define input boundary points in the small scale grid by defining discrete grid points as such. The output of ***ww3_grid*** will echo the corresponding locations if this output is requested at compile level.
- Add these grid points as output grid points in ***ww3_grid.inp*** for the large scale run and run ww3_grid to assure that data can be generated.

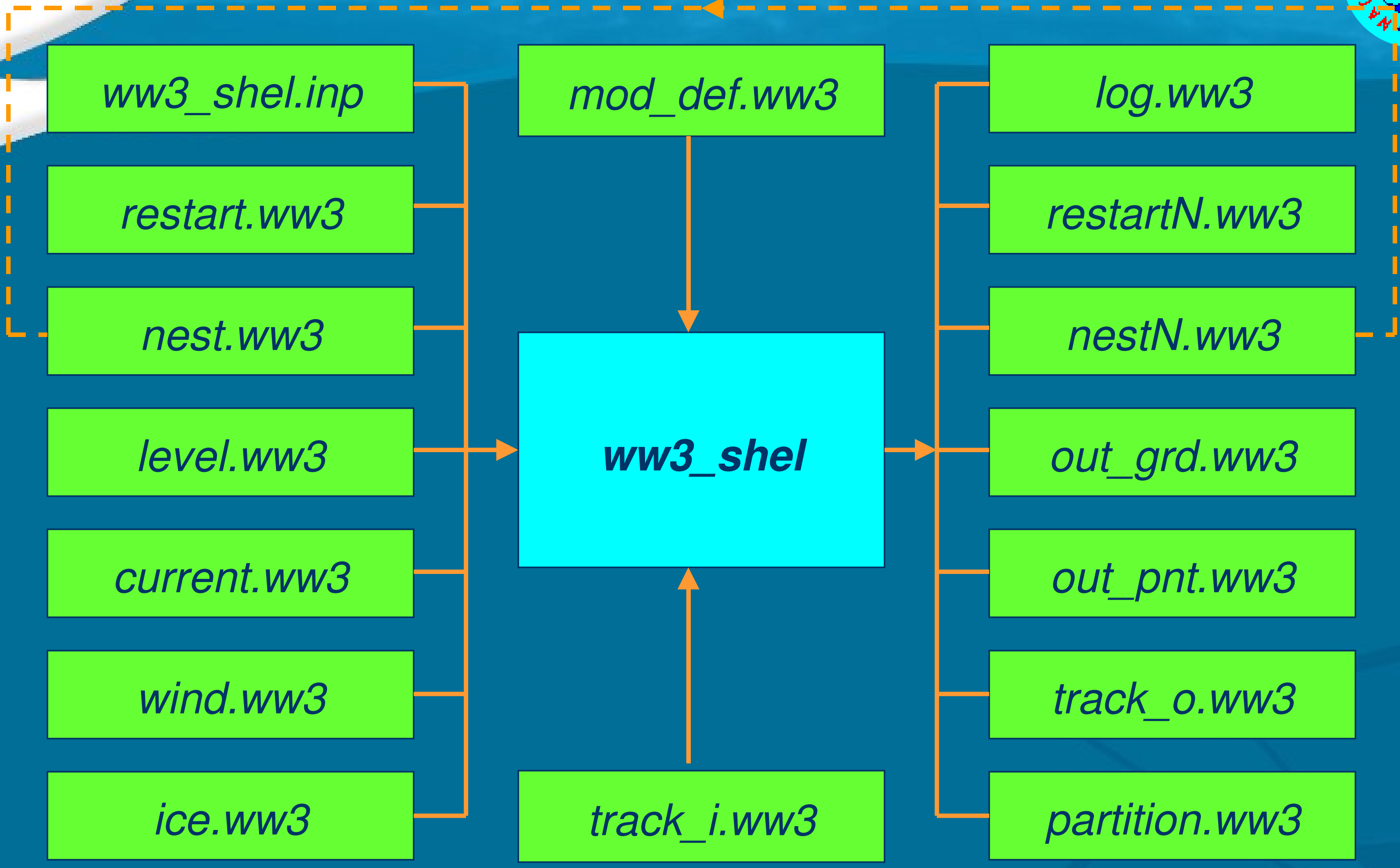
Nesting set-up cont-d :

- Switch on boundary data output in ***ww3_shel.inp***, for the large scale model and run this model for a given time frame.
- Copy the file ***nest1.ww3*** as generated by the large scale model as ***nest.ww3*** to the work directory of the small scale model, and run the small scale model for (part of) the time frame.
- Check ***log.ww3*** of the small scale model to assure that new boundary data was read at the appropriate times

Comments about nesting :

- The model matches up the locations in the file with the locations marked as input point in the grid. The order in which the data points are defined in either model therefore is immaterial.
- The locations of the in and output points may differ by a fraction of the grid size of the small scale grid.
- The data file contains on grid point spectra of the large scale model. The file size can thus be minimized by lining up the input points in the small scale model with the grid in the large scale model.
- Input points for the small scale grid have to coincide with sea points in the large scale grid.

Nesting (ww3_shel)



Beware for the following when using example input files as templates. :

- Make sure time steps are set properly for your application to assure economy and stability. Considerations for the four time steps are:
  - Maximum time step for propagation for lowest frequency waves. CFL criterion needs to be satisfied for stable model computations.
    - Time steps for higher frequencies are automatically adjusted.
    - Currents are automatically accounted for.
  - Overall time step can be bigger, but not too much.
  - Refraction time step should be half overall time step or smaller.
  - Minimum source term time step can be very small.

Pitfalls, cont'ed:

- The test output time step for restart files is set to 1s. If the range of output times is increased without changing this, the model will practically grind to a halt on I/O.

- In the example the garden sprinkler correction for option PR2 is switched of by setting the swell age to something very small. Make sure you switch this on be defining a realistic but stable 'swell age'

  Better to use PR3 to begin with.

- Note that many compilers are not very graceful about running out of memory …..

Multi-grid or mosaic model capability was introduced in model version 3.14

- Full two-way interaction between grids with variable resolution makes a mosaic of grids into a single wave model.
- Detailed description of development and algorithms will be presented in lecture wwws 3.2.
- Here we will concentrate on details of running the model (***ww3_multi***) only.
- Key for successful mosaic is consistency between grids.

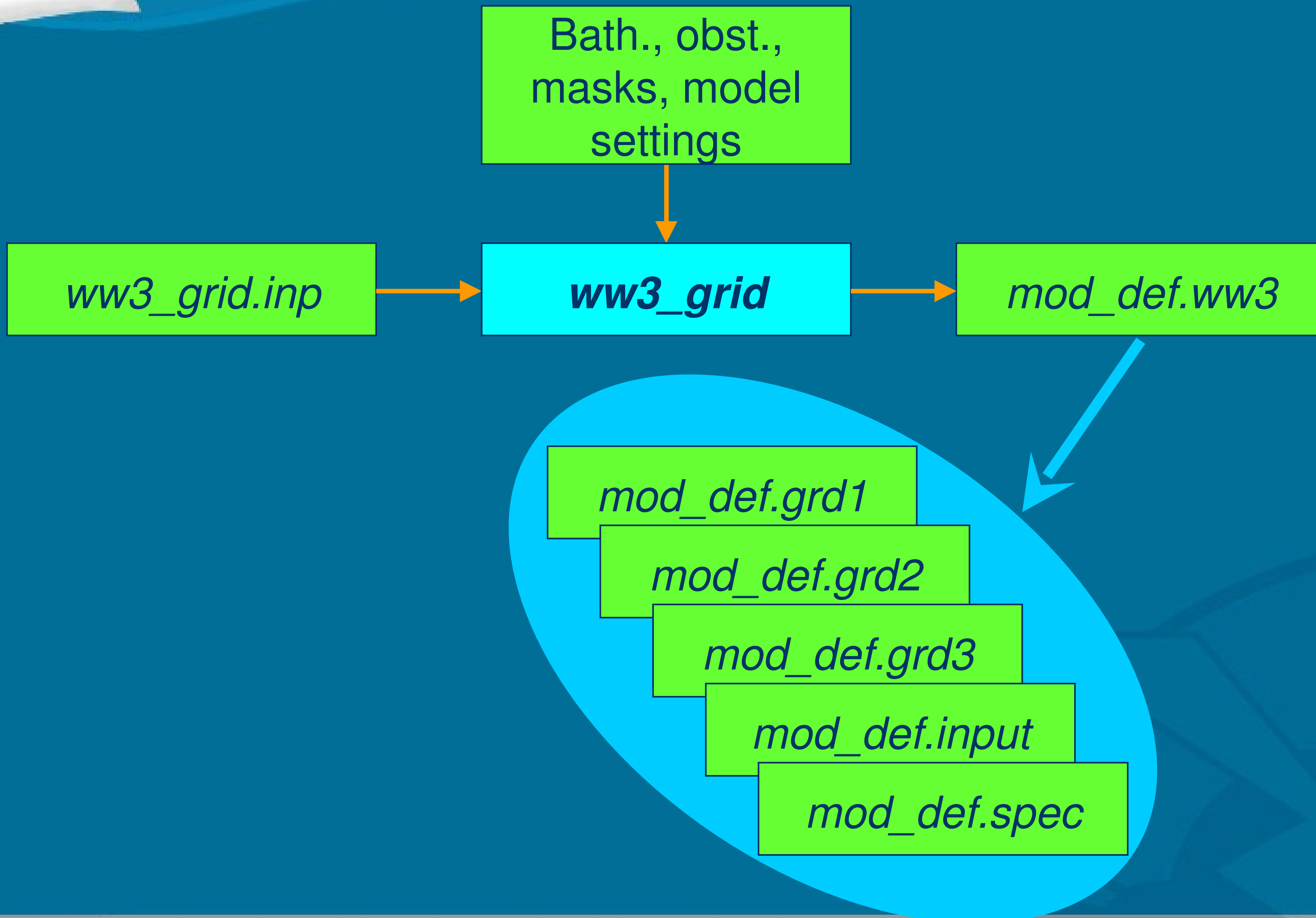
  Automated grid generation recommended.

  See lecture wwws 1.2.

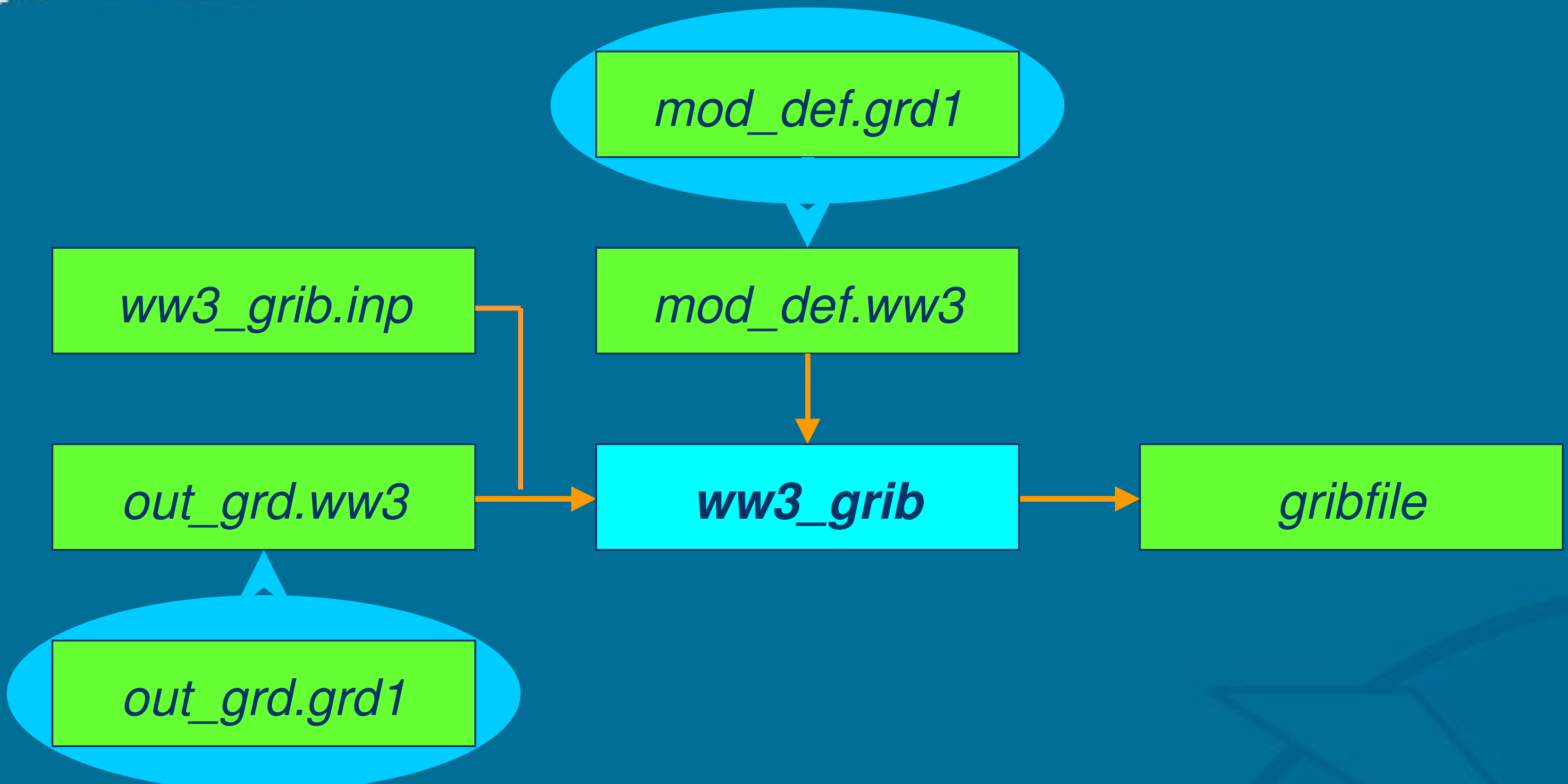
ww3_multi basics:

- ***ww3_multi*** replaces ***ww3_shel*** only, all other programs are used as before:
  - Each individual grid or data for each individual grid is processed as in the single grid model version, however,
  - ***ww3_multi*** identifies each grid by a unique file extension replacing "***.ww3***" in traditional ***ww3_shel*** model applications.
    - Copy preprocessed files to unique file name after processing, e.g., ***mod_def.ww3*** becomes ***mod_def.grd1***.
    - The opposite copy or link is needed to post-process raw model output data.
  - See following examples …..

Mosaic models



Mosaic models



ww3_multi basics cont'ed:

- ***ww3_multi*** recognizes three types of grids:

  Regular model grids identifying grids that make up the mosaic.

  Separate grids on which model input is defined.

  Grids for the actual mosaic can get their input as preprocessed on their own grids (***mod_def.grd1*** using e.g., ***wind.grd1***), or from separate input grids with internal interpolation/averaging within the model. (optional)

  Point output can be produced per grid, or as a single unified file. ***ww3_multi*** does not require spectral grid to be identical throughout (unlike ***ww3_shel***), requiring an extra grid definition. (optional)

ww3_multi basics cont'ed:

- There are many ways to run ***ww3_multi***, here an example will be given of the data flow for the following model set up:

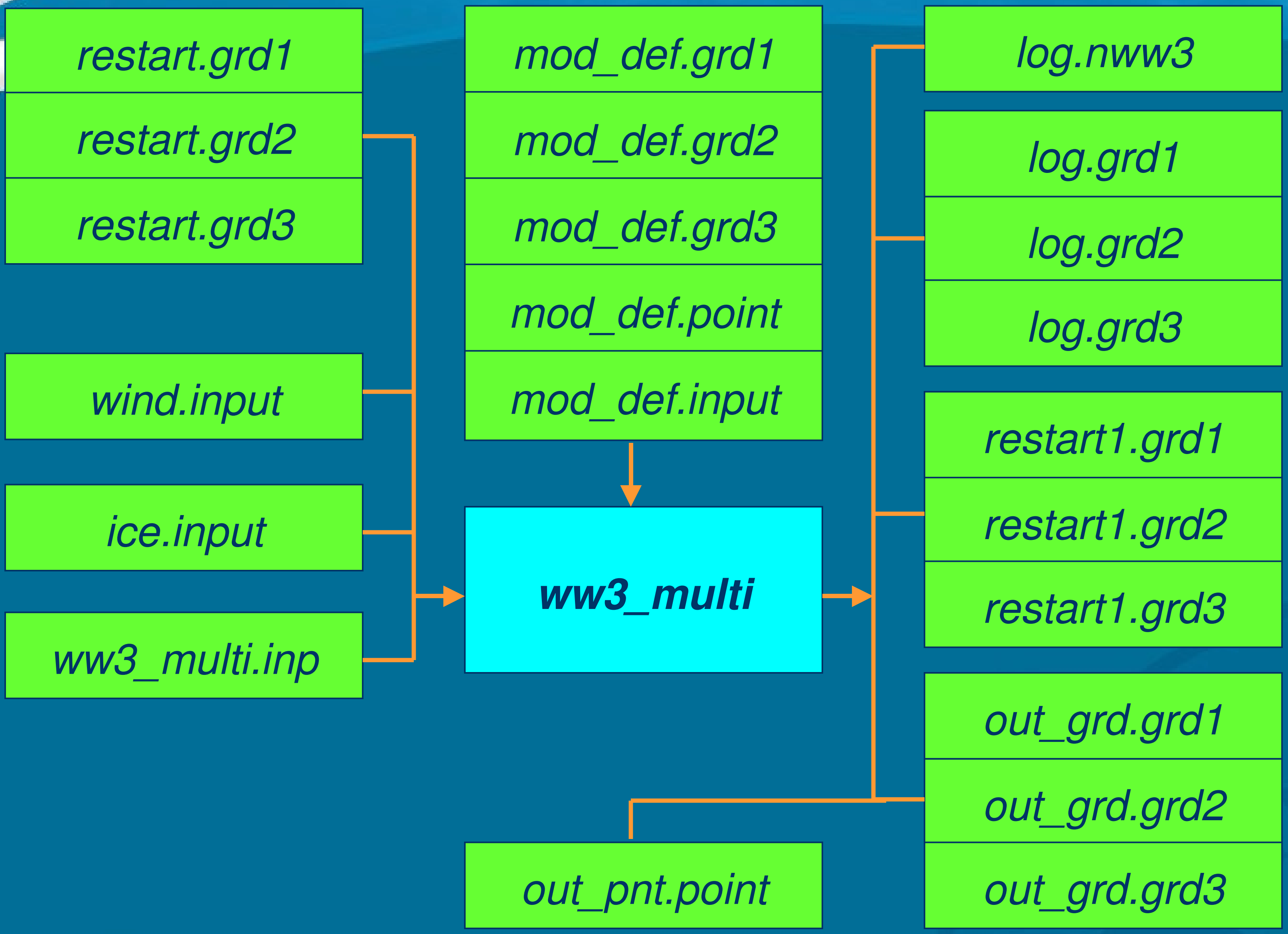
  Three grids in the mosaic, identified with ***.grd1***, ***.grd2***, and ***.grd3***.

  All grids are drive by ice and wind from a common grid ***.input***.

  All grids create their own ***restart*** and ***out_grd*** files, but all point output is written to a single file (***.point***).

- See manual and test cases on how the details of the setup of ***ww3_multi*** work.
- Both idealized and real-work test cases are provided and can be used as templates.

Mosaic models



ww3_multi basics cont'ed:

- The mosaic model needs to know the relationships between grids. For this, a grid rank number is introduced.
  - Grids with (near-)identical resolution get identical ranks.
    - Grids are reconciled in overlap region.
  - Grids with lower resolution get lower rank.
    - Two-way nesting between grids.
- Grids also need a group number. Grids in the same group can be computed side-by-side in a parallel environment.
- More details are given in lecture wwws 3.2.

ww3_multi basics cont'ed:

- For each grid in the mosaic, the following information is needed (all set in ***ww3_grid.inp***).
  - A full grid with proper computational mask.
  - Input boundary points at which the grid expects to get data from lower ranked grids.
  - Grid time stepping information.
- Additional information provided per grid in ***ww3_multi.inp***:
  - Grid rank and group number.
  - Sources of input for grids.
  - Output per grids (or unified point output).
  - Load balancing information.
  - Output processor information if so required.

ww3_multi nesting notes:

- The lowest ranked grids in the mosaic can receive external boundary data from the file ***nest.modID***.
- The same is true for other grids, but the data has to come either from file or from the mosaic, a mix is not allowed.
- There is an option in ***ww3_multi.inp*** to dump a nesting file ***nest.modID*** for individual grids, containing all nesting data used in the mosaic.

  This can be handy for later re-running of individual parts of the mosaic.

  Note that the name of the **input** rather than **output** nesting file is used here.

  By default, internal nesting data is never saved to file.
- Each grid in the mosaic can produce traditional nesting files ***nestN.modID*** as in ***ww3_shel***.

ww3_multi hints and pitfalls

- Build grids up from low resolution to high resolution.
  - Consistency between input points for higher ranked grids and grid masks for lower ranked grids can be tricky.
  - Automated grid generation recommended (see lecture wwws 1.2).

- Properly designed grids can be “plug-and-play”, if
  - Grids are added at consistent higher ranks,
  - Overlapping grids with same rank are introduced simultaneously.
  - Otherwise, adding or removing individual grids may result in incompatibilities between grids in mosaic.

hints and pitfalls cont'ed

- Time steps can be introduced independently per grid, however, for more transparent model (profiling) behavior, it is recommended that

  Grids with identical rank have identical overall time steps.

  Time steps for different grid ranks are integer multiples.

  Note that neither are requirements.

- On parallel (MPI) file systems, first build the mosaic with all grid using all processors. Optimize later by assigning individual grids with same rank to sub-sets of processors, and by designation I/O processors.

hints and pitfalls cont'ed

- The test ***mww3_test_01*** through ***mww3_test_05*** present a wide range of different uses of ***ww3_multi*** using idealized test cases. Some interesting cases are:
  - Providing lateral boundary data for a 2-D model by nesting it in a 1-D model with identical down-wave resolution.
  - Two groups of three grids with identical rank, with two-way nesting between both groups.

- The test cases ***mww3_case_01*** and following represent real world test case that should be suitable as a blueprint for other real-world applications.

The end

End of lecture wwws 1.1